目录

[前言基础 2](#_Toc98430203)

[DOM 插入JS代码方式 3](#_Toc98430204)

[弹出的输入输出语句 弹窗 3](#_Toc98430205)

[变量声明的一些问题 var 4](#_Toc98430206)

[函数 4](#_Toc98430207)

[typeof 检测变量类型 4](#_Toc98430208)

[转义字符 5](#_Toc98430209)

[字面量 表面看出来是啥 5](#_Toc98430210)

[数据转换 6](#_Toc98430211)

[字符型数据转换toString 6](#_Toc98430212)

[转化为数字类型 parse 6](#_Toc98430213)

[转换为布尔型 Boolean() 6](#_Toc98430214)

[数组 6](#_Toc98430215)

[函数 function 7](#_Toc98430216)

[形参和实参个数不匹配 7](#_Toc98430217)

[内置对象 arguments的使用 8](#_Toc98430218)

[作用域 有局部全局变量之分 没有块级作用域 8](#_Toc98430219)

[预解析 8](#_Toc98430220)

[JavaScript 对象 9](#_Toc98430221)

[创建对象的三种方式 调用和遍历 9](#_Toc98430222)

[JavaScript 内置对象(容器) p157 11](#_Toc98430223)

[常用对象(容器) 11](#_Toc98430224)

[Date 日期对象的使用 年 月 日 星期 时间P163 12](#_Toc98430225)

[创建数组的两种方式 13](#_Toc98430226)

[检测是否为数组和翻转,排序 13](#_Toc98430227)

[添加和增加 13](#_Toc98430228)

[获取数组元素索引 13](#_Toc98430229)

[将数组转化为字符串等其他常用方法 14](#_Toc98430230)

[连接 截取删除 14](#_Toc98430231)

[字符串----字符串具有不可变性 14](#_Toc98430232)

[字符串对象的基本包装对象 14](#_Toc98430233)

[Js会把基本的数据类型包装成复杂的数据类型 15](#_Toc98430234)

[有关数据的包装知识 15](#_Toc98430235)

[字符串对象根据位置返回字符或ascll 15](#_Toc98430236)

[字符串常用方法 重要：concat(拼接字符串) slice(start,end) (截取字符串) 16](#_Toc98430237)

[替换字符串和转化为数组,大小写转换 16](#_Toc98430238)

[数据类型的要素 16](#_Toc98430239)

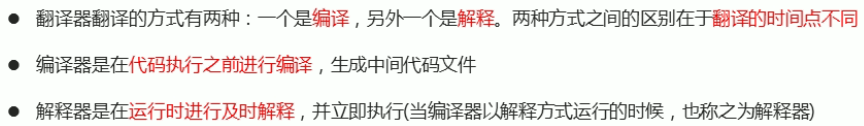
[不同数据类型的存储情况 16](#_Toc98430240)

[var timer = null ; timer 其类型是object 空的对象 16](#_Toc98430241)

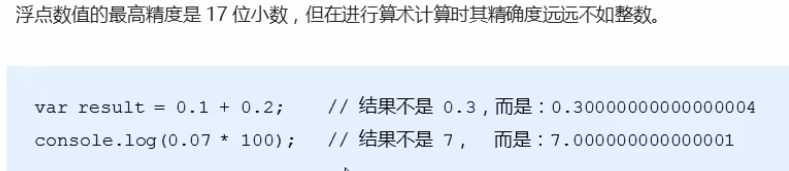
[函数调用时，不同数据类型的变化 17](#_Toc98430242)

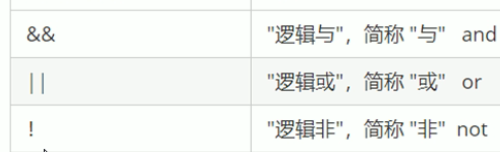
[Web APLS简介 第二阶段 18](#_Toc98430243)

**JS 解释型语言**

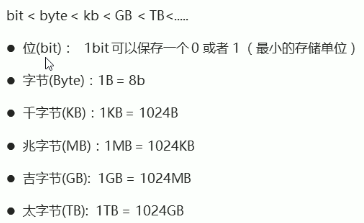


**浮点数会有精度的为问题 不能拿浮点数进行直接比较**





# 前言基础





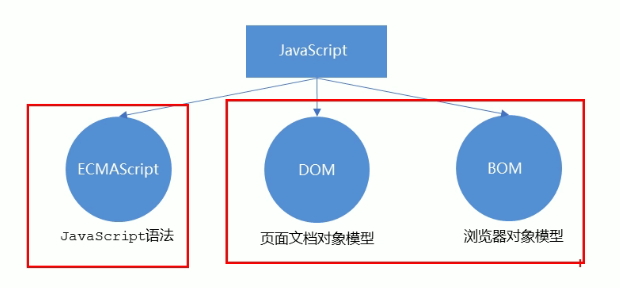
JavaScript下·脚本语言—逐行解释

用于实现业务逻辑和页面控制(决定功能)

浏览器分成 渲染引擎和JS引擎

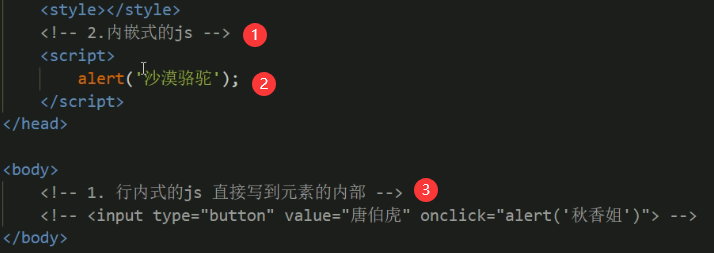
alert( ‘ 弹出 ’ )

JS的组成





# DOM 插入JS代码方式



# 弹出的输入输出语句 弹窗 alert

alert(msg) 弹出警示框

console.log(msg)



# 变量声明的一些问题 var

var age;//声明变量 (variable 可变的)

可以不声明类型直接使用 但会有其他问题，如变成全局变量

命名规则



JS是动态语言 既可以中途更换数据类型

var x=10;

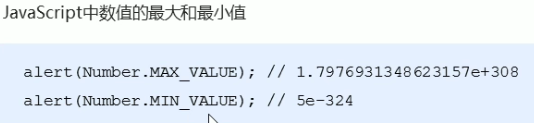
x=’string’;

字符串与其他数值相加，结果变成字符串

lnfinity 无穷大 -lnfinity无穷小 NaN，Not a nimber 非数值

undefined 声明变量没有给值 Null 设置为空

前面加0 表示八进制 0x表示十六进制

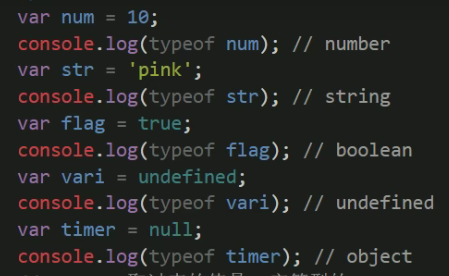


# 函数

IsNaN()判断值是否为数值类型

str.length 获取字符串，数组等的长度

# typeof 检测变量类型

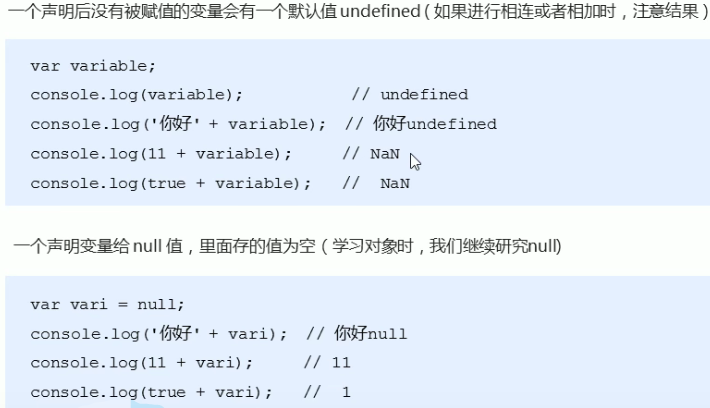


# 转义字符

‘你是“猪”吗‘ 或 “你是’猪’吗”



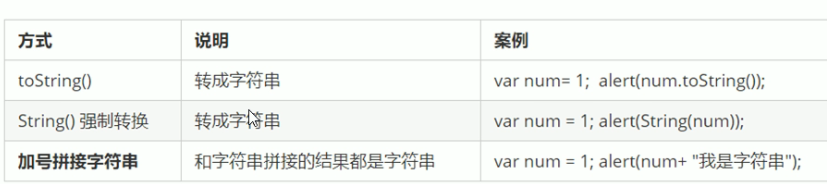
undefined 声明变量没有给值 Null 设置为空



# 字面量 表面看出来是啥

# 数据转换

字符型数据转换toString



转化为数字类型 parse



转换为布尔型 Boolean()

Boolean()函数

代表空值和否定的值会被转换为false ，如 ‘ ’ ，0 ，NaN ，null ，undefinde

其余值被转换为true

        //布尔型的转换

        console.log(*Boolean*('')); //false

        console.log(*Boolean*(0)); //false

        console.log(*Boolean*(NaN)); //false

        console.log(*Boolean*(null)); //false

        console.log(*Boolean*(undefined)); //false

        console.log(*Boolean*('小白')); //false 应该是true

        console.log(*Boolean*(12)); //false 应该是true

# 数组 new Array()

**逻辑中断**

**switch(“判别式”){**

**case value1:**

**“表达式”**

**break;**

**}**

**创建数组： 数组中的类型可以是任意的**

**var arr = new Array();**

**var arr = [ ];**

**var arr = [1,2,’pink’, 3];**

**输出 数组名 是打印整个数组**

**修改数组长度 arr.length = 5; 默认未给值 是undefined empth 空**

**给数组名赋值 arr = ‘？？？’; 数组内的值会全部被替换**

# 函数 function

声明方式一：

function 函数名( 形参 ) { }

声明方式二： ( 函数表达式-匿名函数)

var 变量名=function ( ) { } ; // 使用变量来存储函数

调用时：

函数名(实参)

形参和实参个数不匹配

若是多余的变量不参与运算，则不会影响结果



内置对象 arguments使用“*argument-伪数组-“*论据”

是对于当前域的函数来讲（重点）

        var fd = function(*pi1*, *pi2*) {

*this*.pi1 = *pi1*;

*this*.pi2 = *pi2*;

            console.log(*arguments*[3]);

            console.log(*arguments*.length);

*this*.fang = function() {

                console.log(*Math*.max(*pi1*, *this*.pi2));

                console.log(*arguments*[0]);

                console.log(*arguments*.length);

            }

}

存储了所有传过来的实参

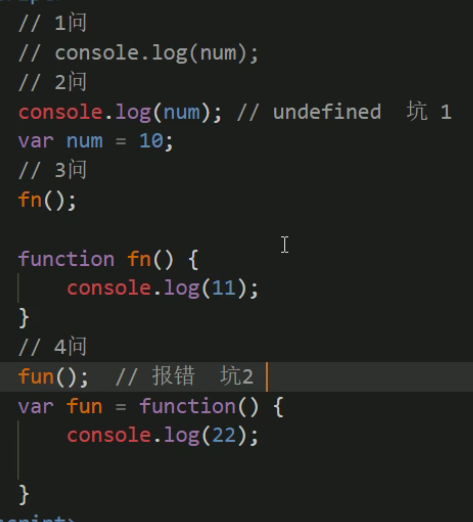
使用了伪数组作为存储 并不是真正意义上的数组，没有数组的一些方法

具有length 和 索引

作用域 有局部全局变量之分 没有块级作用域

if(判别式){块级作用域 声明变量，花括号外可以使用}

**预解析**



JS引擎运行js分为两步： 预解析 代码执行

预解析 js引擎会把js里面所有的var 还有function 提升到当前作用域的最前面

代码执行 按照代码书写顺序从上至下执行

------只是把var的声明提到当前作用域最前面 所以上述中原因是

未赋值 var fn=10 ;

未将函数声明 var fu=function( ) { } ;

**声明不会重复声明 (即该变量被声明为变量后再声明函数是不管用的)**

**赋值会重复赋值 (会覆盖原值)**

# JavaScript 对象

创建对象的三种方式 调用和遍历

**创建对象的第一种方式：**

利用对象字面量创建对象 { }

var obj ={ } ; 创建空对象

对象中的属性不需要声明 直接赋值

**创建对象的第二种方式: new**

var obj = new Object( ) ; 创建了一个空对象

obj.name = ‘ 张三丰 ‘ ； 给对象里面添加属性

obj.fun =function( ){ } ; 添加函数

**创建对象的第三种方式：使用了 构造函数 ( 类似于类 )**

优点:可以同时创建多个对象

function 构造函数名( ) {

this.属性 = 值

this.方法 = function( ) { }

}

        function fun(*pi1*, *pi2*) {

*this*.pi1 = *pi1*;

*this*.pi2 = *pi2*;

*this*.fang = function() {

                console.log(*Math*.max(*pi1*, *this*.pi2));

            }

*this*.fang2 = function(*p1*) {

                for (var key in *p1*) {

                    console.log(key + ":");

                    console.log(*p1*[key]);

                }

            }

        }

**构造 及 调用方式：** new 构造函数名( ) ；

        var p1 = **new** fun(24, 42);

        p1.fang();

        p1.fang2(p1);

**遍历对象属性：for ( 变量 in 对象 ) { } 是类 不是函数**

**之下的 p1是构造函数(仿类) 会将类中属性和函数都遍历**

*this*.fang2 = function(*p1*) {

                for (var key in *p1*) {

                    console.log(key + ":");

                    console.log(*p1*[key]);

                }

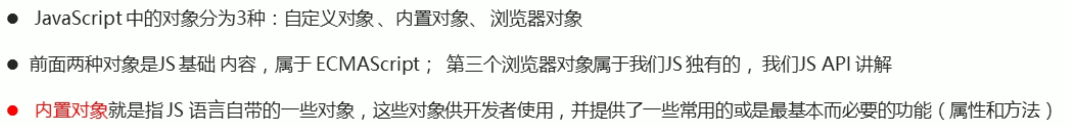
            }

**构造函数的“类”也可以装进变量之中**

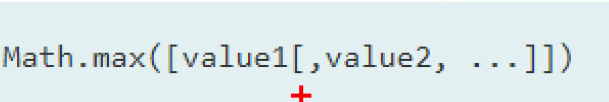
var fd = function(*pi1*, *pi2*) {

# JavaScript 内置对象(容器) p157

文档查阅 https://developer.mozilla.org/zh-CN/



MDN 和 W3C 查阅文档



方(中)括号 表示 写也可以 不写也可以

[developer.mozilla.orq](https://developer.mozilla.orq/zh-CN/)

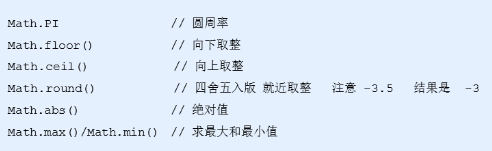
math 对象 它具有数学常数和函数的属性和方法。不是一个函数对象 不是构造器

常用对象(容器) random随机

圆周率 向下取整 向上取整

四舍五入(遇到点五时往大了取，既是负点5时需要注意的)

绝对值 求最大 最小值



随机数方法 Math.random()；返回一个随机区间 [0,1)

以此做一个函数 在确定区间内取任意值 包含两侧值



# Date 日期对象的使用 年 月 日 星期 时间P163



         var obj = **new** *Object*(); //创建一个对象实例

        //1.如果Date没有参数 则返回当前系统的当前时间

        var date = **new** *Date*();

        console.log(date);

        //2.参数常用写法 数字形2019，10，01 或者是字符串型 '2019-9-9 8:8:8'

        var date1 = **new** *Date*(2019, 10, 1);

        console.log(date1);

        var date2 = **new** *Date*('2019-9-9 8:8:8');

        console.log(date2);

        console.log('\*\*\*\*\*\*\*\*\*我是分界线\*\*\*\*\*\*\*');

        //返回当前日期的年 2019

        console.log(date.getFullYear() + '年');

        //获取月份  月份由0到11表示  故需要加1

        console.log(date.getMonth() + 1 + '月');

        //几号

        console.log(date.getDate() + '号');

        //星期几  返回的是星期  周日是0  因为周日是第一天 同月份一样

        console.log('周' + date.getDay());

        //可以使用 字典 存储中文的周日等

        var arr = ['星期日', '星期一', '星期二', '星期三', '星期四', '星期五', '星期六'];

        console.log(arr[date.getDay()]);

        //日期格式化 时 分 秒

        console.log(date.getHours());

        console.log(date.getMinutes());

        console.log(date.getSeconds());

        //获取毫秒数(时间戳)  不是当前的毫秒数 是1970年一月一日至今的毫秒数

        console.log(date.valueOf()); //获取毫秒方式一

        console.log(date.getTime()); //获取毫秒方式二

        var daet3 = +**new** *Date*(); //获取毫秒方式三 (常用)

        console.log(daet3);

        console.log(*Date*.now()); //获取毫秒方式四

数组

创建数组的两种方式

第一种 字面量方式

var arr=[1,2,3,4,5];’

第二种 new Array()创建

var arr = new Array(2); //创建数组长度为2

var arr = new Array(2,3…); //里面参数大于2 即为是数组里面的序列

检测是否为数组和翻转,排序

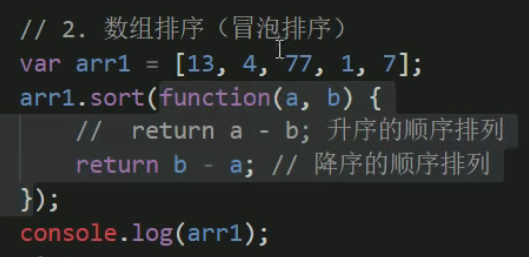
arr Instanceof 检测 ； Instanceof相当与一个运算符//返回bool型

Array.isArray(检测) ；//返回bool型

reverse() //会改变原数组 并返回新数组

sort() //会改变原数组 并返回新数组

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



添加和增加

push(参数1...) 末尾添加一个或多个元素返回新的长度

pop() 删除数组最后一个数组 返回删除值

nushift(参数1..) 向数组开头添加一个或多个元素 返回新的长度

shift() 删除数组的第一个元素 返回删除的元素值

获取数组元素索引

arr.indexOf(查找元素); 从左向右找到第一个 若数组中无该元素 返回-1；

arr.lastIndexOf(查找元素); 与上相同(但从右向左查找)

arr.indexOf(查找元素 , 起始); arr.lastIndexOf(查找元素 , 起始);

将数组转化为字符串等其他常用方法



连接 截取删除

join分隔符默认是逗号，可以自己修改。



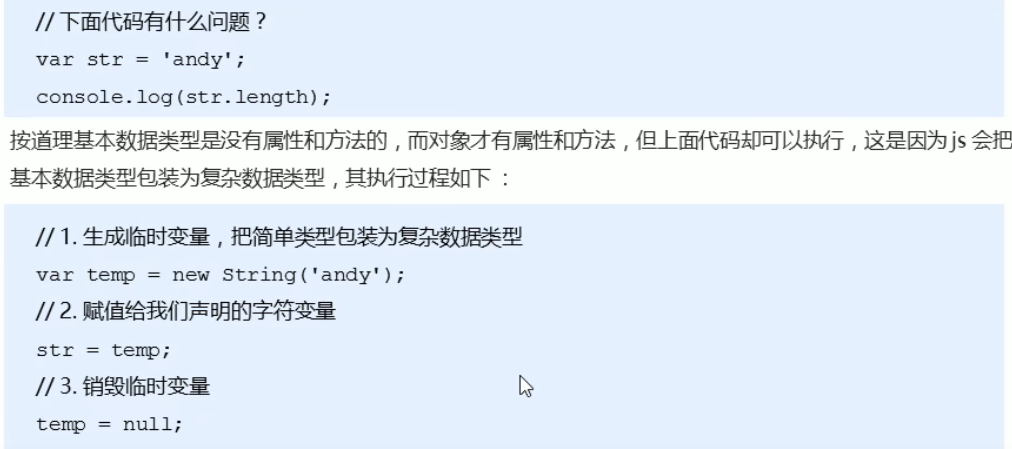
# 字符串----字符串具有不可变性

**字符串具有不可变性,即在运行过程中，更改字符串的值，其分配的内存并没有被销毁，而是重新分配内存，存储新的值，再指向其内存地址**

字符串对象的基本包装对象

**对变量声明赋值时的一个处理，与其他无关 下面既是解释**

# Js会把基本的数据类型包装成复杂的数据类型



有关数据的包装知识

包装数据类型包括 String Number Boolean;

这三个基本类型都有自己对应的包装对象。包装对象，其实就是对象，有相应的属性和方法。调用方法的过程，是在后台偷偷发生的，所以我们称作为基本包装类型。

1.存储的时候按基本数据类型类型存储,

2.使用的时候转化为复杂数据类型的形式让你使用,

3使用完之后又瞬间变成基本数据类型存储.

作用:可以使用对象的方法;

eg:str.toString()

字符串对象根据位置返回字符或ascll



p183

字符串常用方法 重要：concat(拼接字符串) slice(start,end) (截取字符串)

**(start，length) 也是取出字符串中一段 截取起始后 length长度**

**方法 方法 方法 方法 方法 需要调用 这才叫做方法**



替换字符串和转化为数组,大小写转换

toUpperCase()转换为大写

toLowerCase()转换为小写

replace(‘被替换的字符’ , ‘替换为的字符’)

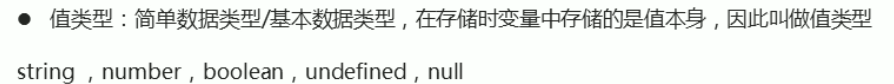
split(‘分隔符’) 将字符串转换为数组 与上面的转换相对 join

# 数据类型的要素

不同数据类型的存储情况

简单类型 又叫 基本数据类型 或者 值类型

放于栈(只是方便理解)

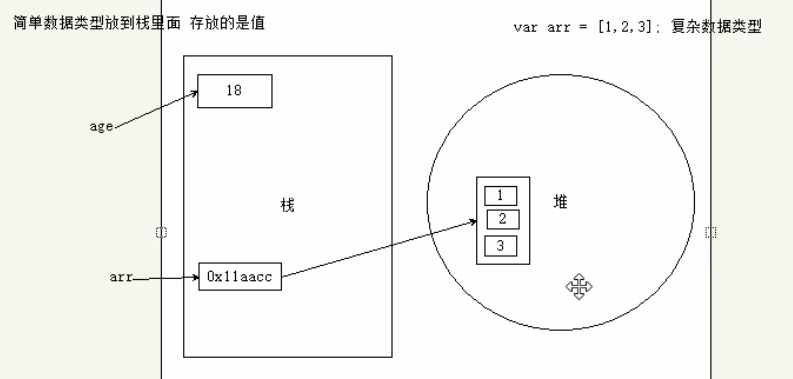


var timer = null ; timer 其类型是object 空的对象

使用环境 若有变量想以后存储为对象，暂时还未想好存储物品，这个时候给null

复杂类型 又叫做 引用类型 (需要使用new )

放于堆中(只是方便理解)



函数调用时，不同数据类型的变化

**●函数会有若干个接受的形参，**

**●传入简单类型时，形参接受到的是数据，**

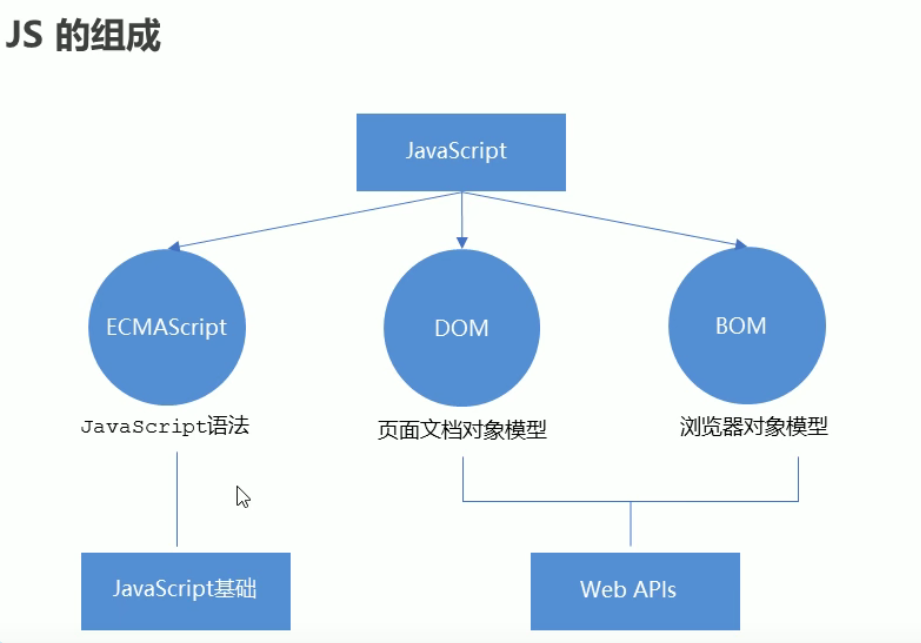
**●传入复杂类型时，接受到的是地址。**

**下面是详解！**

**●**简单类型传参，其数据只在栈中，传入函数是将数据传入，函数内部的更改不会影响原数据

**●**复杂类型传参，因为其在栈中存储的是地址，传入函数中的也会是地址，函数类操作是直接访问堆中数据的

# Web APLS简介 第二阶段





P192 结束